LANGUAGE DOCUMENTATION & CONSERVATION

Vol. 7 (2013), pp. 114-122
http://nflrc.hawaii.edu/ldc
http://hdl.handle.net/10125/4576

Ukelele

From SIL International

Reviewed by Tyler M. Heston, *University of Hawai'i at Mānoa*

1. INTRODUCTION. Ukelele is a tool for creating and customizing software keyboards that is developed by SIL International.[1] It is released under SIL's Freeware License.[2] This review is based on version 2.2.4, installed on a Macintosh MacBook Pro running OSX 10.8.3.

I begin the review by presenting a brief background of the problem that Ukelele helps to solve, as well as some other related software tools. The main section of the review covers the primary features of Ukelele and its use. I conclude with some recommendations regarding the use of Ukelele.

2. HOW TO TYPE PHONETIC SYMBOLS? One constant problem that linguists have had is how to type special phonetic characters. This problem predates the advent of computers, as the difficulty of typing special characters was also present with typewriters. When writing by hand, it is easy to write any character imaginable; however, with typewriters and early computer technology, users were limited to a restricted character set. In spite of this, developments in computer technology have made it possible to type even complicated sequences of phonetic symbols with relative ease. While the following discussion focuses mostly on the matter of typing phonetic symbols, such as the characters of the IPA, most of the points below are also relevant to all interested in typing characters beyond the letters, numbers and punctuation marks used in standard English orthography.

In examining the issue of typing special characters, it is helpful to establish some background on how computers represent and manipulate characters in general. By character, I mean any letter, number, or symbol. At the deepest level, computers represent everything as a binary number, that is, a sequence of ones and zeros; characters are no exception. What was needed was a standard for mapping basic numbers and symbols to binary numbers that could be stored by a computer. One standard that was developed for this purpose in the early years of the 1960s is ASCII (usually pronounced ['æski]), which stands for "American Standard Code for Information Interchange."[3]

The ASCII standard was—and still is—very important, however, it is limited in that it only represents a basic set of 128 different characters. This basic set includes lowercase

---

[1] The copyright for Ukelele is held by John Brownie, SIL. Ukelele can be downloaded for free at: http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=Ukelele. The information about Ukelele in the following review is drawn from the manual, tutorial and help menus that are bundled with the program, as well as the program itself.

[2] http://www-01.sil.org/computing/catalog/freeware.html

[3] http://www.w3schools.com/tags/ref_ascii.asp

E-ISSN 1934-5275

and uppercase Latin letters, a few control sequences for special operations and the punctuation marks and symbols that one can see on a typical keyboard.[3] In order to represent other symbols and other writing systems, another standard was needed. While there were many other standards developed for individual writing systems, the major breakthrough came in the early 1990s with the advent of the Unicode standard.[4] Unicode provides a unique identifying number for an extremely large number of letters, characters and symbols. While the original ASCII set could only represent 128 different characters, the current version of Unicode supports thousands of characters: scientific symbols, phonetic symbols, logographic symbols, and letters and numbers from a wide range of writing systems.[5]

Using Unicode is tremendously beneficial and a great help to linguists using phonetic symbols in computer applications. However, even with Unicode, a problem still remained. Even though Unicode provided a standard way to represent an astronomically large range of characters, computer users were mostly limited to using the same types of keyboards that had been common before the development of Unicode. Furthermore, given the breadth of Unicode, developing a keyboard with an individual key for every symbol in the standard would be impracticable. What computer users needed was a practical way to type Unicode characters without special hardware.

There are a number of tools that have been developed to allow users to input Unicode characters. One type of tool is a *character map utility*, which is a small program that presents a user with a window containing a large number of different Unicode characters. The user can then select a character to copy or drag to their desired location. Simple character map utilities are often bundled with operating systems (e.g. Macintosh's Character Viewer). In addition, there are a large number of character maps available online, such as the IPA picker at http://people.w3.org/rishida/scripts/pickers/ipa/.

Another type of tool is a *software keyboard*. A software keyboard is a tool that changes which character is entered when a given key on a keyboard is pressed. Of particular interest here are those keyboards that allow users to type special characters by pressing certain combinations of keys. Using key combinations to access additional characters beyond those that are usually available on a standard keyboard is a particularly powerful tool, because it greatly expands the range of what can be typed.

A third type of tool is a *keyboard editor*. A keyboard editor allows an individual to create and customize their own software keyboards so that they can decide which keystrokes will result in a given character. Ukelele is one such utility for Macintosh OSX. Ukelele presents a simple user interface for designing and editing a software keyboard. Ukelele then translates a user's decisions about the layout of the keyboard into an XML keyboard layout file that is readable by the Macintosh operating system. The intended audience is primarily linguists who use Macintosh computers and who would like to be able to customize the software keyboards they use.

---

[4] The following information about Unicode is taken from http://www.unicode.org/

[5] For more information on Unicode, fonts, and encodings, see The Unicode Consortium 2012, Korpela 2006, Haralambous 2007, Perry 2010.

3. USING UKELELE.

3.1. STARTING A PROJECT IN UKELELE. When opening Ukelele for the first time, one is presented with a blank keyboard, as shown in Figure 1. While this may be a good place to start if one is interested in designing a completely new and unique software keyboard, it is often advantageous to begin a project based on an existing keyboard that is similar. This reduces work, because it is possible to use a keyboard that is already made and only change the things that will be different. Ukelele can import '.keylayout' files, and open them for modification. Ukelele also comes with a large number of software keyboards in this format, which are available for modification. Another option is to create a new keyboard based on the keyboard that is currently in use. For making a software keyboard for IPA symbols, it is beneficial to start with a standard QWERTY keyboard and make modifications from there. For non-Roman keyboards, it is best to start with whatever keyboard is closest to the desired finished product.

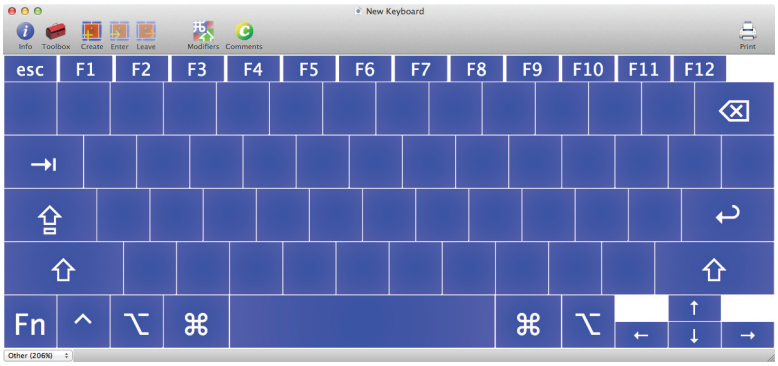

FIGURE 1: A new project

To start a new project based on an existing software keyboard layout file, choose File > New Based On. If the window does not default to the folder containing the pre-defined keyboards, it may be necessary to navigate there manually. When it is first downloaded, the Ukelele application is enclosed in a zipped folder called 'Ukelele,' which contains other resources. The pre-existing software keyboards can be found in the folder 'System Keyboards,' in the folder 'Ukelele' after it is unzipped. The folder should be located wherever it was originally downloaded.

The pre-defined keyboard 'U.S. Extended' is a good base for new projects. This keyboard follows the layout of standard U.S. keyboards, and makes many accent marks available by using the option key in combination with other letters. The 'U.S. Extended' keyboard can be found in the folder 'Unicode' under 'System Keyboards.' To start a new project based on this keyboard, choose File > New Based On, navigate to the file 'USExtended.keylayout,' and click 'Open.' A new project should open, with a window that looks like Figure 2. The window will show which symbol each key on the keyboard will produce. To see what character will be produced if the shift key is held in combination with another letter, hold the shift key. The shift key will turn a different color, and the characters on each of the keys will change. The same process is used for other modifier keys, such as option;

holding each of the modifier keys will reveal the association between characters and keys when that modifier is pressed.

FIGURE 2: New project based on the U.S. Extended keyboard

At this point, we will need to set the name of the keyboard. That way, the new keyboard we are creating will be distinguishable from the original 'U.S. Extended' keyboard, which will be unaffected by the modifications that we make. The name should be short and descriptive. Now is also a good time to save our file. At this point, it does not matter where we store the file, although once we are finished, we will need to drag it into a special location on the hard drive so that the computer will recognize it. It is helpful give the file the same name as the keyboard. The file will have the extension .keylayout, which identifies the file as a software keyboard. Click 'Save.'

3.2. MODIFYING KEYBOARDS. At this point, it is possible to begin modifying the values of specific keys simply by double clicking the key and changing the value in the textbox that appears. However, in all likelihood, when designing a keyboard for inputting IPA characters, we will not want to modify the basic set of keys. As IPA uses ASCII symbols in addition to the special IPA characters, we will want to make a way to access the special characters without changing most keys' normal values. One way to do this is to create what is called a 'dead key.' A dead key is a key that does not enter a character by itself, but rather, changes how the next key is interpreted. So, for instance, if we made the semicolon a dead key, pressing the semicolon key on the keyboard would not automatically enter a semicolon. Instead, pressing the semicolon key would change what would show up when the next key was pressed. For instance, on an IPA keyboard, typing ';e' might result in the character 'ɛ.' Using a dead key thus sacrifices one key (since that key no longer has its usual value), however, it makes it possible for each of the other keys to have a different value than normal, thus allowing us to use them to represent an entirely new range of symbols. The next few paragraphs demonstrate how to create a dead key, following the steps outlined in the tutorial bundled with the program.

To create a new dead key, click 'Keyboard > Create Dead Key.' After you do so, the message space across the bottom of the Ukelele window will say 'Press or click the new dead key.' Following the instructions, either click on the key that you would like to become

a dead key with the mouse or press the appropriate key on the keyboard. Upon doing so, a dialog box will appear, asking you to enter a name for the dead key state. When a software keyboard is in a 'dead key state,' it means that a specific dead key has just been entered, and the following key that is pressed may have a special value. If you are adding a large number of dead keys to the same keyboard, it might be helpful to have descriptive names for the state created by each of the different dead keys, but if you are only adding one or two dead keys, the default names should be sufficient. Click 'OK.' Next, Ukelele will ask for a terminator for the dead key, which is the character that will be displayed if the user types the dead key, and follows it with a character that does not have a defined value when preceded by that dead key. For most purposes, it suffices to leave this field blank. Click 'OK' again.

The dead key has now been created, and you are now viewing what values each of the keys on the keyboard will have when entered after the semicolon dead key. You should see a keyboard layout that is blank except for a few modifier keys, such as control and shift. Suppose that we want the character 'ɛ' to show up when we type the sequence ';e.' To do this, we first find the 'e' key; it might be helpful to press the 'e' key on the keyboard and see which key in the Ukelele window changes color.

There are a number of ways to associate the symbol 'ɛ' with the 'e' letter key. One way is to use the built-in Macintosh character map utility, and simply drag and drop the character from the character map directly onto the desired key in the Ukelele window. Another way is to double-click the key in Ukelele. Double-clicking creates a text box into which you can enter the desired character. If you happen to have another IPA keyboard utility installed, you can simply type the character into the text box. If not, you can copy and paste or use a character map utility. You can also type the Unicode codepoint, prefixed with '&#' and followed by a semicolon. For instance, to get the character 'ɛ', which is associated with the decimal number 603 in the Unicode standard,[6] you could type 'ɛ' into the text box.

Regardless of the way the character is entered, you should now see the character 'ɛ' over the appropriate key in Ukelele. The process for filling the rest of the keyboard is the same. For each key, you can chose which character it should be associated with. It is also possible to combine dead keys with shift. For instance, typing the sequence ';e' will have a different outcome than ';E.' To modify the shifted version of key, simply press and hold the shift key, and modify the value of the key as before.

To exit the dead key state and view the mapping between key and character when the dead key is not pressed, click 'Keyboard > Leave Dead Key State'. The dead key that was entered—in this case, the semicolon—will be red, as shown in Figure 3. To re-enter the dead key state, click 'Keyboard > Enter Dead Key State.' There will be a drop-down menu, from which you can choose the name of the dead key state you would like to enter. One may continue adding keys and dead keys following these same steps, remembering to save periodically. Figures 3 and 4 illustrate a simple keyboard that uses the semicolon as a dead key to access several common IPA symbols. Alternatively, if there is a dead key in another software keyboard that already has the desired functionality, it is possible to import that dead key, with the associated dead key state, directly into the current keyboard layout. This

[6] http://www.unicode.org/charts/

function can be found under 'Keyboard > Import Dead Key.' The built-in help utility has more information about importing a dead key.

FIGURE 3: A simple keyboard that uses the semicolon as a dead key to access a number of frequently-used IPA and related characters. Keyboard is in its regular state.

FIGURE 4: The same keyboard as Figure 3, in dead key state

3.3. LOADING KEYBOARDS INTO THE OPERATING SYSTEM. While you are working on a keyboard, the changes you make do not immediately take effect. This is convenient, because if you accidentally remove the letter 'e' from the keyboard you are working on, you can simply re-enter the letter using the keyboard, rather than having to locate it in a character map utility. All the changes that you make, rather, are stored in an XML file created by Ukelele, which ends in the file extension '.keylayout.' The file follows Macintosh OSX's XML syntax for software keyboards. In order to start using the new keyboard we have developed, it needs to be saved in a place where the operating system will recognize it and enable it for use.

The first step is to place the '.keylayout' file we have created in a place where OSX will recognize it. There are a few different locations that will work, though the easiest place for me has been to put the keyboard in the folder '/Library/Keyboard Layouts', that is, in the folder 'Keyboard Layouts', which is within the folder 'Library', which is at the top level of the hard drive. The folder 'Library' is a special folder where the operating system

keeps many of the important files that are essential for it to function properly. For this reason, if you try to drag your '.keylayout' file directly into '/Library/Keyboard Layouts', you will probably be greeted by an error message such as this one (from OSX 10.8.3): "The item 'Demo.keylayout' can't be moved because 'Keyboard Layouts' can't be modified." In OSX 10.8.3, this message is accompanied by two options, 'Authenticate' or 'Cancel'. At this point, it would be good to check that you are moving the correct file to the correct location, and if so, click 'Authenticate', type your password, and hit enter. The file should now appear in the folder.

The second step is to tell the operating system that the keyboard layout we have just created is one that we would like to use. Open System Preferences, click 'Language and Text' (or 'International', depending on the version of OSX) and choose 'Input Sources'. There should be a long list of country flags and language names. These are all of the software keyboards that are available for you to use. Clicking on the check box next to the name of a keyboard will activate that keyboard.

Find the name of the keyboard that you just created and click the check box next to its name. If the new keyboard does not appear in the list, you may need to log out and log back into your computer so that it recognizes the new keyboard. On my machine, running OSX 10.8.3, logging out was not necessary for a newly added keyboard, although the tutorial for Ukelele, which appears to have been created with an earlier version of OSX in mind, states that logging in again is necessary. Also, if you see a choice such as 'Show Input menu in menu bar', make sure that box is checked.

The third step is to choose the newly created keyboard from the 'Input menu' in the 'menu bar' at the top of the screen. One my machine, the Input menu looks like a small United States flag along the top of the screen, towards the right, near the time, battery and volume icons. Clicking on the input menu reveals a list of each of the software keyboards that have check marks by them in the 'Language and Text' pane of System Preferences. To change keyboards, simply select a different keyboard from the list. From this menu, you can choose between whichever keyboard you were using before and the keyboard you have just created. At this point, you can test your keyboard in any word-processing program. If you find any changes you would like to make, you can make them in Ukelele, resave the file, remove the file from the Keyboard Layouts folder, and follow the three steps given above to reinstall the keyboard.

4. RECOMMENDATIONS. I would recommend Ukelele for tech-savvy linguists who would like the control to create their own keyboard layouts. In addition to creating keyboards for entering IPA characters, Ukelele could be very helpful for developing software keyboards for orthographies with non-ASCII symbols. For instance, if the orthography for a language used a 'barred i' (ɨ) for a high central vowel, but did not use all the consonants on an English keyboard, it would be relatively easy to make a new keyboard for that language that used one of the unneeded consonant keys on an American keyboard for ɨ, which would avoid the need for difficult keystrokes. Ukelele could also be very useful for those working on languages that use non-Roman scripts. For users that are only interested in typing the IPA and who have less need for fully customizable keyboards, SIL also has a

few different IPA software keyboards that are already completed and ready for download from their website.[7]

Although Ukelele is not strictly necessary, as one could in principle write the XML for a '.keylayout file' in any text editor, having a graphical user interface that automatically generates the correct XML is tremendously helpful. Unfortunately, since Ukelele is designed to generate an XML record of the mappings between keystrokes and characters in the format used by OSX, it is only available on Macintosh computers. There are other, similar utilities available for Microsoft Windows, such as the non-free Keyman application.[8]

Ukelele fulfills a very particular niche in the toolkit of the language documenter, and thus, many important concerns about best practices in data management and data preservation are less relevant in the analysis of Ukelele. In my opinion, the main way that Ukelele supports best practices is by supporting Unicode, which is a widely accepted standard for character representation. Using Unicode supports data accessibility, because data encoded in Unicode is much more likely to be broadly accessible than data encoded in a more obscure character encoding scheme, both now and in the future (for more about best practices, see Bird & Simons 2003, Gippert 2006, Gibbon et al. 2004).

Unfortunately, by its very nature, Ukelele is quite dependent on the proprietary Macintosh OSX operating system in order to function properly, and it is therefore not as portable as one might desire. If the keyboard layout used by Apple becomes more widespread in the future, Ukelele will of course become correspondingly more portable, however, for the present time, Ukelele seems to be limited to the Macintosh operating system. One positive is that even though Ukelele is somewhat dependent on proprietary software in order to function properly, it does not lock any language data into a proprietary format. Even if the XML keyboard layout formats used by Ukelele were to become completely defunct in the future, language data that was entered using a software keyboard developed with Ukelele will not be any less accessible, because it is encoded using the Unicode standard.

In conclusion, Ukelele is a very specialized tool that does its job very well. If there are no premade keyboards available that satisfy your needs, and you are using OSX, I would highly recommend using Ukelele.

---

7 For a list of Macintosh, Linux and Windows keyboards available from SIL, see the following: http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=UniIPAKeyboard

8 http://www.tavultesoft.com/keymandev/

Primary function	Creating and editing custom software keyboard layouts
Pros	It is easy to learn, intuitive to use, allows for creation of complex keyboard layouts, allows for easy modification of existing layouts, promotes native integration into the OSX keyboard layout system, supports Unicode, and outputs the final product into an XML format that can be modified by advanced users if needed.
Cons	Is restricted to the Macintosh OSX operating system, and installing a created keyboard layout is somewhat complicated.
Platforms	Mac OSX only
Open source	Unknown
Proprietary	No. Ukelele is freeware.
Reviewed version	2.2.4
Application size	12.7 MB
Documentation	A manual (Brownie 2012), tutorial, and built-in help menu are included in the download. There is also a 'Ukelele Users group' on Google Groups.

References

Bird, Steven & Gary Simons. 2003. Seven dimensions of portability for language documentation and description. *Language* 79(3). 557–582.

Brownie, John. 2012. *Ukelele: Unicode Keyboard Layout Editor Version 2.2 manual*. 2012. Online: http://scripts.sil.org/cms/scripts/page.php?item_id=ukelele

Gibbon, Dafydd, Catherine Bow, Steven Bird & Baden Hughes. 2004. Securing interpretability: the case of Ega language documentation. *Proceedings of the 4th International Conference on Language Resources and Evaluation. European Language Resources Association: Paris*, 1369–1372. Online: http://www.lrec-conf.org/proceedings/lrec2004/pdf/138.pdf

Haralambous, Yannis. 2007. *Fonts & Encodings*. (Trans. by) P. Scott Horne. Sebastopol, CA: O'Reilly Media, Inc.

Korpela, Jukka. 2006. *Unicode Explained*. Sebastopol, CA: O'Reilly Media, Inc.

Perry, David J. 2010. *Document Preparation for Classical Languages: Latin, Greek, Biblical, and Medieval*. 2nd ed. Greentop Publishing.

The Unicode Consortium. 2012. *The Unicode Standard, Version 6.2.0*. Mountain View, CA: The Unicode Consortium. Online: http://www.unicode.org/versions/Unicode6.2.0/

Tyler Heston
theston@hawaii.edu